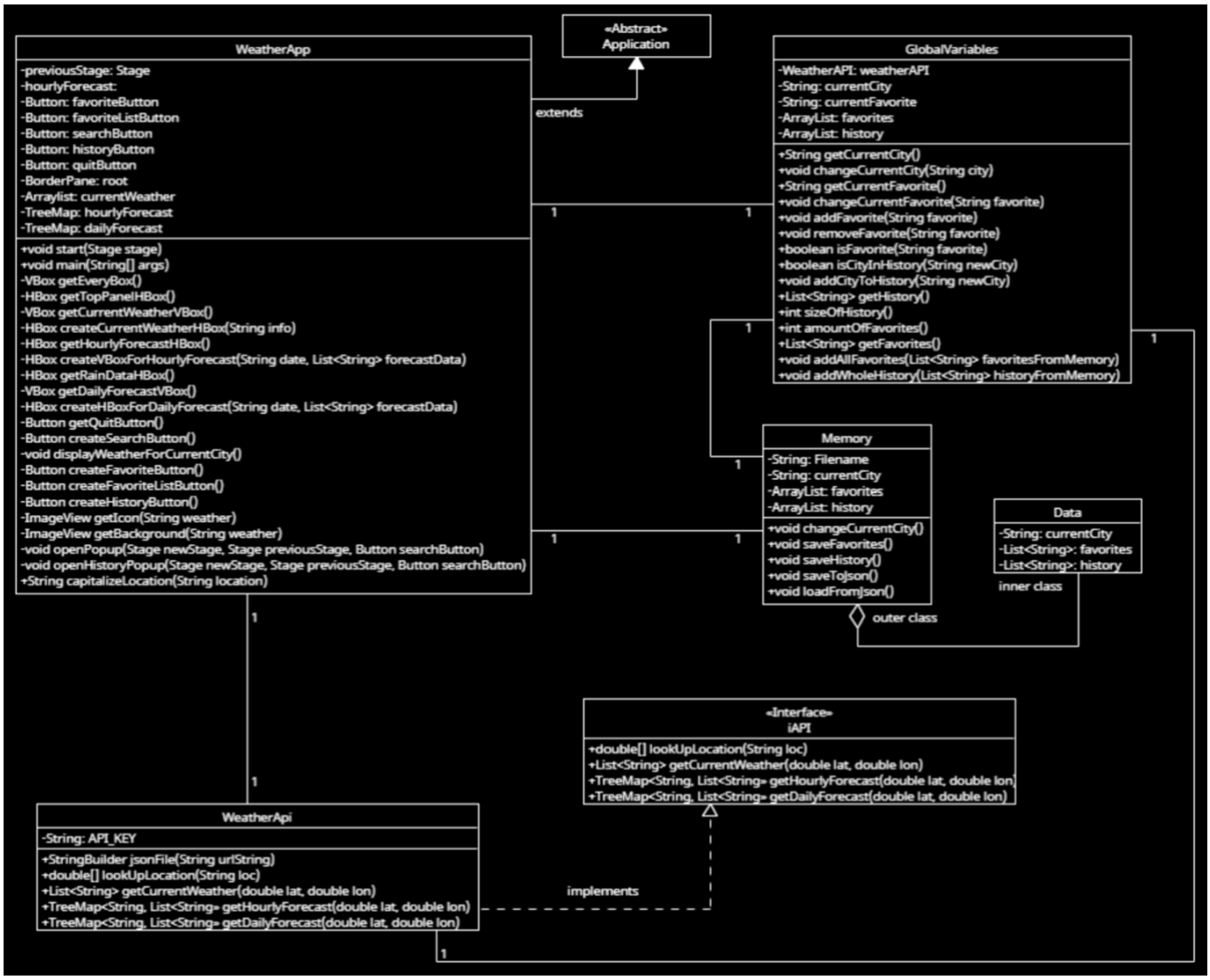
UML kaavio



Tärkeimpien luokkien vastuujako

Ohjelmassa on käytännössä neljä tärkeintä luokkaa, WeatherApp, WeatherAPI, GlobalVariables ja Memory. WeatherApi luokassa haetaan tarvittavat tiedot OpenWeatherMap API:ssa. Tämän luokan tekemiseen osallistui kaikki ryhmän jäsenet. WeatherApi luokka sisältää pääasiassa GUI:n toiminnan mahdollistavat toiminnallisuudet. Tämän luokan tekemisessä Piitu oli suurimmassa vastuussa. Matias ja Liisa ovat myös lisäillyt pienempiä osia WeatherApp:n toteutukseen. WeatherApi luokassa haetaan tarvittavat tiedot OpenWeatherMap API:sta. Tämän luokan tekemiseen osallistui kaikki ryhmän jäsenet. GlobalVariables luokka pitää sisällään suosikkien ja historian toiminnan. Tämän luokan tekemisestä vastuussa oli Matias. Memory luokka mahdollistaa json tiedostoon tallentamisen ja siitä lukemisen, kun ohjelma suljetaan ja avataan uudestaa. Tämän luokan toteutuksesta vastuussa oli Liisa.

Ohjelman toiminta

Ohjelma aloitetaan tavallisesti käynnistämällä ohjelma tietokoneelta haluamallaan tavalla. Ohjelman ensimmäisen käynnistyksen jälkeen näytölle avautuu graafinen käyttöliittymä, joka ei vielä näytä mitään säätietoja. Nyt käyttäjä voi etsiä säätietoja haluamastaan sijainnista kirjoittamalla sijainnin tekstikenttään ja klikkaamalla ”Search”, jonka jälkeen säätiedot esitetään käyttöliittymässä. Jos etsitään paikkaa, jota ei löydy OpenWeatherMap:sta, käyttöliittymässä ei tapahdu mitään.

Jokaisesta sijainnista näytetään aina tämänhetkinen sää, tuntikohtainen sääennuste seuraavalle 12 tunnille ja päiväkohtainen sääennuste seuraavalle seitsemälle päivälle. Tämänhetkisessä säässä näytetään tämänhetkinen lämpötila, kuvaus säästä ikonilla, päivin minimi ja maksimi lämpötilat ja tuulen nopeus. Tuntikohtaisessa säässä näytetään kellonaika. Tämä aika on aina UTC +3 ajassa. Sen lisäksi ennusteessa on säätä kuvaava ikoni ja tunnin keskilämpötila. Päiväkohtaisessa säässä näytetään päivämäärä, säätä kuvaava ikoni ja päivin minimi ja maksimi lämpötila. Tämän lisäksi käyttöliittymän taustalla on kuva, joka kuvaa sijainnin tämänhetkistä säätilaa.

”Search” napin oikealla puolella on nappi, jossa lukee “<3”. Tätä painamalla käyttäjä voi lisätä tällä hetkellä näytetyn sijainnin suosikiksi. Sen jälkeen napin teksti muuttuu näyttämään “</3”. Tätä painamalla sijainnin voi poistaa suosikeista. Tämän napin oikealla puolella on “favorites” nappi, jota painamalla avautuu ikkuna, jossa näytetään kaikki suosikit. Painamalla jotain suosikkisijainneista voi etsiä kyseisen paikan säätiedot. Suosikki-ikkunan voi sulkea siis painamalla jotain suosikeista, jolloin haetaan uusi sääennuste tai painamalla “quit” tai “x”, jolloin käyttöliittymän sääennuste ei muutu.

Favorites napin oikealla puolella on “History” nappi, jota painamalla aukeaa samanlainen ponnahdusikkuna kuin suosikki-ikkuna, mutta suosikkien sijasta siinä näytetään lista maksimissaan 10 viimeksi haetusta paikasta. Historia-ikkuna toimii muuten täysin samalla tavalla kuin suosikki-ikkuna.

Tuntikohtaisen ennusteen alapuolella on “Get Rain Data” nappi, jota painamalla aukeaa ikkuna, jossa näytetään kuvaaja seuraavan 12 tunnin sademääristä.

Kaikista oikeinpuoleisin nappi on “Quit” nappi. Sitä painamalla käyttöliittymä sulkeutuu. Toinen tapa sulkea käyttöliittymä on sulkea ikkuna sen kulmassa olevasta “x” napista.

Kun käyttöliittymä avataan uudestaan, se näyttää sääennusteen paikasta, joka oli etsitty viimeksi, ennen kuin käyttöliittymä suljettiin. Tämän lisäksi “Favorites” nappia painamalla löytyy nyt suosikit, jotka sinne on lisätty edellisillä käyttökerroilla ja “History” napin alla näkyy maksimissaan 10 viimeisintä paikkaa, joiden ennusteita on haettu. Käyttöliittymässä on siis muisti, joka tallentaa suosikit, nykyisen sijainnin ja historian levylle, kun ohjelma suljetaan ja lataa ne ohjelmaan uudelleen käynnistämisen yhteydessä.

Ohjelmassa on siis toteutettu seuraavat vapaaehtoiset toiminnot, jotka on kuvattu aiemmassa tekstissä:

- Ohjelmaan on toteutettu omatoimisesti graafinen käyttöliittymä. Voit käyttää pohjana valmiina annettua projektia, mutta tällöin käyttöliittymässä on oltava selkeää omaleimaisuutta annettuun pohjaan verrattuna.

- Ohjelmalla voidaan tallentaa paikkakuntia suosikeiksi, jolloin niiden ennuste voidaan helposti hakea myöhemmin uudelleen.
- Ohjelman tila (nykyinen paikkakunta sekä suosikit) tallennetaan ohjelman sulkeutuessa levylle, josta ne ladataan ohjelman uudelleen käynnistämisen yhteydessä.
- Yksinkertaisen ennusteen sijaan näytetään yksityiskohtaisempi ennuste, johon kuuluu sekä tuntikohtainen ennuste että päiväkohtainen koonti, josta löytyy vähintään päivän alin ja ylin lämpötila.
- Ohjelma käyttää omia ikoneita (ei säädatapalvelusta haettuja). Ikoneilla on oltava enemmän variaatiota, eli useampia erilaisia, kuin mitä säädatapalvelu tarjoaa.
- Ohjelma selviää hallitusti tiedostojen käsittelyssä tapahtuvista virheistä.
- Ohjelmalle on toteutettu yksikkötestejä.

Joista kaksi ovat lisäominaisuuksia:

- Kuvaajien käyttö. Ohjelmassa näytetään kuvaajia hyödyntäen jotain dataa, kuten vaikkapa lämpötila tai sademäärä.
- Paikkakuntien hakuhistoria. Viimeksi haetut paikkakunnat pidetään muistissa, jotta niihin on helppo myöhemmin palata. Myös hakuhistoria tallennetaan tiedostoon ja palautetaan, kun ohjelma käynnistetään uudelleen.

Työnjako

Aloitimme työnteon sillä, että sovimme, minkä tasoisen toteutuksen haluamme tehdä. Sen perusteella päätimme, mitä ominaisuuksia tavoitteenamme on toteuttaa. Tämän jälkeen sovimme, että aina kun jollain ryhmän jäsenellä on aikaa koodata, seuraavan tehtävän voi valita vapaasti tehtävistä, jotka ovat vielä tekemättä. Tarkempaa tehtävänjakoa emme päättänyt ennen ohjelmoinnin aloittamista.

Toteutunut työnjako meni suunnitelmien mukaisesti. Konkreettisesti suunnitelma toteutui siten, että Matias teki kokonaan GlobalVariables luokan ja siihen liittyvät lisäykset WeatherApp:iin. Liisa teki Memory luokan ja siihen liittyvät lisäykset WeatherApp:iin. Piitu teki suurimman osan GUI toteutuksesta ja kaikki osallistuivat WeatherAPI:n tekemiseen. Tarkemmin toteutunut tehtävänjako näkyy GitLaib:sta

Käyttöohje

Ohjelma aloitetaan tavallisesti käynnistämällä ohjelma tietokoneelta haluamallaan tavalla. Säätietoja voi etsiä kirjoittamalla halutun sijainnin nimi tekstikenttään halutulla kielellä ja painamalla ”Search” nappia. Kaupungin voi lisätä suosikiksi ”<3” napilla ja poistaa suosikeista samalla napilla. Tällöin napissa lukee ”</3”. Suosikkeja voi katsoa ”Favorites” napista ja niiden sääennusteen saa painamalla haluttua paikannimeä listasta. Historian etsiminen toimii muuten samalla tavalla, mutta nyt täytyy painaa ”History” nappia. Sademääräkuvaajaan saa avattua painamalla ”Get Rain Data” nappia. Käyttöliittymän voi sulkea painamalla ”Quit” nappia tai painamalla käyttöliittymän yläkulmassa olevaa ”x” nappia.

Tiedossa olevat ongelmat ja puutteet

Merkittävin asia, joka ohjelmasta puuttuu, on ajan muuttaminen oikeaan aikavyöhykkeeseen. Ohjelma näyttää ennusteet aina suomen aikavyöhykkeen mukaan. Tämä tarkoittaa sitä, että jos etsii esimerkiksi New Yorkin säätä, tuntikohtaisessa sekä päiväkohtaisessa sääennusteessa ajat ovat suomen ajassa eikä New Yorkin ajassa.

Toinen puute on se, että suosikeissa ja historiassa ei tarkisteta, että kaikki sijainnit ovat samalla kielellä. Suosikeissa voi siis olla esimerkiksi London, joka on englantia ja Tukholma, joka on suomea. Kuitenkaan sama paikka ei voi olla kummassakaan listassa kahdesti eri kielillä, koska sijaintien olemassaolo listoilla tarkistetaan koordinaattien mukaan.

Kolmas asia, jota voisi parantaa olisi, että sademääräkuvaajien sademäärän suuruus y-akselilla skaalautuisi sen mukaan, kuinka paljon sadetta on luvattu. Tällä hetkellä y-akseli on aina yhtä suuri.

Ongelmana on myös, että joillekin sääkuvauksille ei löydy ikonia, vaikka katsoimme, että meillä pitäisi olla ikoni kaikille kuvauksille. Silloin GUI näyttää ”?” ikonin.

Tekoälyn käyttö

Kaikki ryhmänjäsenet käyttivät projektissa tekoälyä apuna. Yleinen tapa, jolla käytimme tekoälyä, oli pyytää ideoita siihen, miten joku tietty toiminnallisuus kannattaisi toteuttaa. Esimerkiksi, jos oli joku GUI:hin liittyvä asia, jota emme aikaisemmin ollut tehnyt, kysyimme tekoälyltä mitkä olisivat sen mielestä järkeviä tapoja toteuttaa toiminnallisuus. Toinen asia, johon käytimme tekoälyä, oli ongelmanratkaisu. Jos koodissa oli joku ongelma tai virhe, jonka syytä emme tienneet, kysyimme tekoälyltä, mikä voisi olla ongelman aiheuttava asia. Näillä tavoilla tekoälyä on käytetty hyvin laajasti ohjelmassa.